Project supported by Acorn Computers

ACORN COMPUTER

ME & MY MICRO

Games devised by
Fred Harris

Programming Consultant
Richard Freeman

THE BASIC LISTINGS
for the Acorn Electron and Sinclair Spectrum computers

Yorkshire Television

Dear Microphile

Computing is frustrating, time-consuming, irritating, bewildering, and great fun. There's more satisfaction to be had from developing your own twenty-line BASIC program than from adding a few more megazaps to your latest arcade game score.

Me & My Micro is aimed at the relative (or absolute!) newcomer to programming. It's one way of getting to grips with the micro, by writing simple games. Not that I think games-writing is the ultimate goal of every aspiring programmer – it just happens to be the way I went about tackling BASIC.

Once you can tackle BASIC on your own, you can do your own tax returns, solve second-order differential equations, anything you like. But first, you need to get to grips with your micro to find out how to think to make it work.

What you won't find in this booklet are the most exciting and fast-moving games around. In fact, they are all relatively slow and simple. But what you will find is the detail of how each one is put together. Not only have we used simple games; we have also chosen to use 'structured programs' to make them easier to understand – that is, each program consists of a sequence of self-contained blocks, located away from the main body of the program. The idea of this is to keep everything as clutter-free and readable as possible – unlike 'spaghetti' programs, they should also be easier to 'de-bug'. It's by no means the only way of doing things, and structure is no guarantee of elegance – or indeed a working program! But it might help.

Whatever style of programming you adopt, don't be afraid to experiment.

The games in the series were deliberately chosen to be easy to alter, improve, extend and transform. There are suggestions in these listings and on the TV show. But try out ideas of your own. Whatever happens, you can't damage the micro from the keyboard!

Happy creative computing

FRED HARRIS

P.S. Alternative versions of these listings are available for the Commodore 64, Vic 20, Dragon, Oric, Atari, Sharp 700, and MTX.
If you wish to obtain the above listings please send £1 – including P & P– to:

Computer Training College
Norvic House
1–7 Hilton Street
Manchester M4 1LP

P.P.S.The Electron programs will also run on the BBC micro.

Electron Programs

MONSTERZAP CORE

```
  10 REM MONSTERZAP
  12 REM CORE LISTING
  20 REM Copyright Fred Harris
  25 REM Electron edition: Richard Freeman
  28
  30 MODE 6
  35
  40 REM Initialise
  50 GOSUB 1000
  60
 100 REM Draw scene
 110 GOSUB 3000
 120
 150 REM Main movement loop
 155 REPEAT                                          <-④
 160   FOR c=0 TO 39
 170     PRINT TAB(c,r) "*";                         <-⑨
 180     FOR t=0 TO delay : NEXT t                   <-⑤
 185     LET key$=INKEY$(0)                          <-⑥
 190     IF key$="f" OR key$="F" THEN GOSUB 5000     <-⑦
 200     PRINT TAB(c,r) " ";                         <-⑨
 210     NEXT c
 220   UNTIL FALSE                                   <-⑧
 230
 235 *FX12,0
 240 STOP
 250
 990 REM Initialise
1000 LET r=5                                         <-①
1010 LET f=0
1020 LET delay=25                                    <-②
1060 REM Turn off keyboard auto repeat
1070 *FX11,0                                         <-③
1080 RETURN
1090
2990 REM Draw scene
3000 PRINT TAB(0,10) " HH      HH      HH      HH      HH      HH "
3010 PRINT TAB(0,11) " HH \ / HH \ / HH \ / HH \ / HH \ / HH "
3020 PRINT TAB(0,12) " HH 0*0 HH 0*0 HH 0*0 HH 0*0 HH 0*0 HH "
3030 PRINT TAB(0,13) " HH =&= HH =&= HH =&= HH =&= HH =&= HH "
3410 PRINT TAB(0,18) "0 SHOTS USED"
3420 RETURN
3430
4990 REM Zap                                         <-⑩
5000 FOR l=10 TO 13
5040   PRINT TAB(c,l) " "
5050   NEXT l
5080 LET f=f+1                                       <-⑪
5090 PRINT TAB(0,18);f
6000 IF f=40 THEN STOP
6010 RETURN
```

Main program

Initialise subroutine

Draw scene subroutine

Zap subroutine

Monsterzap Core (Notes)

1. LET r = 5 makes the zapper run across row 5 of your screen (i.e. six lines down). Change this to position the zapper higher or lower than as written in our listing.
2. LET delay = 25 controls the pause between zaps. To slow the program down, set 'delay' to a larger value. To speed it up, try a smaller value. (The Electron computer runs at a slower speed than the BBC Micro so you will need a smaller value for 'delay' on the Electron than on the BBC.)
3. *FX11,0 controls one of the BBC and Electron micro's special effects. It turns off the keyboard 'auto-repeat'. i.e. it re-sets the keyboard so that, on pressing a key, only one character per key press appears on the screen even when the key is held down. To cancel *FX11,0, use *FX12,0.
4. 'c' stands for column. In this loop, varying c moves the zapper across the screen in row 5.
5. FOR t = 0 TO delay : NEXT t is the simplest way of producing a pause but the length of delay cannot be predicted in advance. If you want to produce a delay of exactly n seconds, you should use

   ```
   TIME = 0
   REPEAT
   UNTIL TIME = n*100
   ```

6. INKEY$ (0) takes a keystroke from the keyboard – if there happens to be one. Unlike 'INPUT', INKEY$ (0) does not *wait* for input. If no key is pressed, INKEY$ (0) allows the program to move on to the next statement. The number in brackets relates to the length of time the computer waits for a key press.
7. This line ensures that the zap routine (lines 5000 to 6010) is only used when the F key is pressed.
8. See page 31 for the note on REPEAT loops.
9. This innocent semi-colon is very important on the BBC/Electron. Without it, the print cursor jumps to the next line when the print line is finished. This will either make your display scroll up the screen or leave an ugly flashing cursor somewhere on your screen.
10. This prints a space, so acting as an electronic rubber. Any object overprinted with this is wiped out and replaced by the background colour.
11. f counts the number of zaps that you have used. After each zap, the value of f is increased by 1.

Variables Used

c	column	Controls the column in which the zapper is printed.
r	row	The row in which the zapper appears.
t	time	Counter for the delay loop.
f	fire	The number of zaps used.
delay		Controls the length of the delay.
l	line	Counter for the zap loop.

Suggestions for extending the program
See 'Monsterzap improved'.

MONSTERZAP IMPROVED

```
 10 REM MONSTERZAP IMPROVED
 20 REM Copyright Fred Harris
 25 REM Electron edition: Richard Freeman
 30 MODE 1
 35
 40 REM Initialise
 50 GOSUB 1000
 60
 70 REM Instructions
 80 GOSUB 2000
 90
100 REM Draw scene
110 GOSUB 3000
120
150 REM Main movement loop
155  REPEAT
157   COLOUR 131
160   FOR c=0 TO 39
170     PRINT TAB(c,r) CHR$ 225;
180     FOR t=0 TO 100 : NEXT t
185     LET key$=INKEY$(0)
187     *FX15,0
190     IF key$="f" OR key$="F" THEN GOSUB 5000
200     PRINT TAB(c,r) " ";
210     NEXT c
220  UNTIL FALSE
230
235 *FX12,0
240 STOP
250
990 REM Initialise
1000 LET r=3
1010 VDU 23,224,255,231,231,231,255,231,255,255 :
REM Part of monster
1020 VDU 23,225,129,219,165,153,153,165,219,129 :
REM Zapper
1030 VDU 23,226,36,0,36,255,0,0,0,0 :
REM Part of building
1035 VDU 23,227,0,0,0,0,0,133,137,255 :
REM Dead monster fragment
1036 VDU 23,1,0;0;0;0;
1040 COLOUR 128
1055 REM Turn off keyboard auto repeat
1060 *FX11,0
1070 LET f=0
1080 RETURN
1090
1990 REM Instructions
2000 PRINT TAB(15,3) "MONSTERZAP"
2010 PRINT TAB(12,5) "PRESS F TO FIRE"
2015 PRINT TAB(13,7) "ONLY 40 SHOTS!"
2020 PRINT TAB(4,30) "PRESS RETURN WHEN YOU ARE READY"
```



```
2030 INPUT "" d
2060 CLS
2070 RETURN
2080
2990 REM Draw scene
3000 REM Sky
3005 COLOUR 131
3010 FOR n=0 TO 18
3020   PRINT TAB(0,n) "                                        "
3030  NEXT n
3035
3050 REM Ground
3055 COLOUR 130
3060 FOR n=1 TO 3
3070   PRINT TAB(0,n+18) "                                        "
3080   NEXT n
3090
3095 REM Stars
3100 GCOL 0,1
3105 FOR n=1 TO 50
3110   PLOT 69,RND(1200),550+RND(300)
3120   NEXT n
3125 COLOUR 1
3140
3160 REM Skyscrapers
3162 COLOUR 128
3164 COLOUR 2
3170 FOR n=0 TO 5
3180   FOR l=14 TO 18
3190     PRINT TAB(7*n+1,l) CHR$226 CHR$226
3200     NEXT l
3320   NEXT n
3322
3325 REM Monsters
3327 COLOUR 131
3329 COLOUR 1
3340 FOR n=0 TO 4
3360   PRINT TAB(7*n+3,16) " \ / "
3370   PRINT TAB(7*n+3,17) " 0" CHR$224 "0"
3380   PRINT TAB(7*n+3,18) " " CHR$224 " " CHR$224 " "
3390   NEXT n
3400 COLOUR 0
3410 PRINT TAB(0,22) "0 SHOTS USED"
3420 RETURN
3430
4990 REM Fire
5000 FOR l=14 TO 18
5010   SOUND 1,-10,53,1
5020   PRINT TAB(c,l) "*"
5030   PRINT TAB(c,l) "X"
5040   PRINT TAB(c,l) " "
5050   NEXT l
5060 PRINT TAB(c,18) CHR$ 227
5080 LET f=f+1
5090 PRINT TAB(0,22); f
5100 IF f=40 THEN STOP
5110 RETURN
```

Monsterzap Improved (Notes)

1–4 These lines create special characters (called User Defined Graphics*) using the VDU 23 statement. Any character with an ASCII code between 224 and 255 can be re-programmed in this way. Here, we've chosen to use:

224 as  (Part of a monster)

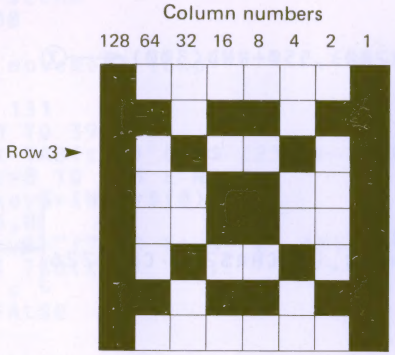
225 as  (The zapper)

226 as  (Part of a building)

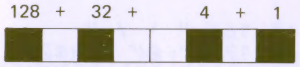
227 as (A dead monster fragment)

As an example, here is how we created the zapper:

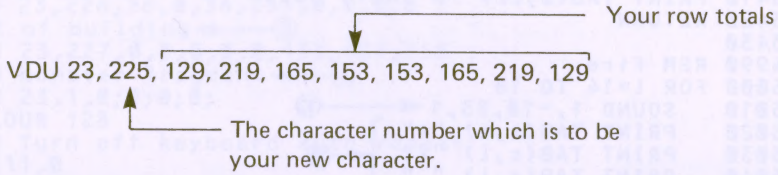
(a) Draw it as blobs on an 8 x 8 grid.



(b) For each row, add up the column numbers of the blocked in columns.

e.g. row 3 

(c) Put all your row numbers into a VDU 23 statement:



5. Once you have defined a special character, you use it by preceding it with CHR$. e.g. to print the zapper, write PRINT CHR$225;. (Note the semi-colon again at the end of the line.)

* For more information on these, see BBC Microcomputer System User Guide pp 170–176 or (Electron ref.)

6. This is a gap of 40 spaces. A neater way of doing this is STRING$ (40, " ").
7. Lines 3105 to 3120 scatter 50 stars about the heavens.
   PLOT 69 produces a dot on the screen when you are using a graphics mode. The full statement must include the position of the dot on the screen in graphics coordinates e.g. PLOT 69, 600, 500 prints a dot near the centre of the screen.

8a & 8b The loop at lines 3180 to 3200 draws six skyscrapers, evenly spaced across the screen; the loop at 3340 to 3390 then draws five monsters between the buildings. This saves an awful lot of typing.

9. INPUT " " d halts the program until the player presses a key. Here 'd' is a dummy variable i.e. the program doesn't use whatever value d has, but we must have a variable in the INPUT statement. CLS (line 2060) clears the screen after a key is pressed, removing the instructions before the skyscraper scene is drawn.
10. This SOUND statement produces a firing noise each time the zap button is pressed.
11. These three characters whiz down the screen from the zapper, giving the appearance of a bomb dropping. Note that the last character to be printed is a space. This makes the bomb appear to drop down the screen.

Forwards and backwards

The Monsterzap (Spectrum version) which you will have seen on television has a zapper which moves left/right then right/left across the screen. This is harder to implement in BBC BASIC but you can do it by changing lines 160 and 210 to

160 FOR I = –39 TO 39
210 NEXT I

and adding a new line

165 c = ABS I

(ABS gives the positive value of a number i.e. ABS 3 is 3 and ABS –3 is also 3.)

Colour

We have introduced colour into this game. In mode 1, we have four colours for foreground (buildings, bombs, etc.) and four for background (sky, earth, water, etc.). These are controlled by colour statements:

	Foreground		background	
Colour	Graphics	Text	Graphics	Text
Black	GCOL0,0	COLOUR 0	GCOL0,128	COLOUR 128
Red	GCOL0,1	COLOUR 1	GCOL0,129	COLOUR 129
Yellow	GCOL0,2	COLOUR 2	GCOL0,130	COLOUR 130
White	GCOL0,3	COLOUR 3	GCOL0,131	COLOUR 131

So, you can see that in the program, we have used:

1040 COLOUR 128		Black background	(text)
3005 COLOUR 131	(Sky)	White background	(text)
3055 COLOUR 130	(Ground)	Yellow background	(text)
3100 GCOL0,1	(Stars)	Red foreground	(graphics)
3162 COLOUR 128	(Skyscrapers)	Black background	(text)
3164 COLOUR 2		Yellow foreground	(text)
3327 COLOUR 130	(Monsters)	Yellow background	(text)
3329 COLOUR 1		Red foreground	(text)
3400 COLOUR 0	(Message)	Black foreground	(text)

Suggestions for improvement

a. Arrange for a bomb to drop from the zapper towards the monsters.
b. Build in a time limit.
c. At the end of the game, arrange for the whole cycle to start again at a higher speed.
d. Build in a penalty for hitting the buildings.
e. *Or* change line 190 to prevent firing when the zapper's over a building.
f. Allow the player to reverse the direction of the zapper. (e.g. Press RETURN to reverse the direction of movement.) Then make the monsters fire back!
g. As the game progresses, lower the zapper's flight row. If the player doesn't zap all the monsters by the time the zapper hits the buildings, he loses.
h. Make an explosion appear on the screen when a monster is hit.
i. Add a deep beep for hitting a wall.

QUACMAN

```
 10 REM QUACMAN
 20 REM Copyright Fred Harris
 25 REM Electron version: Richard Freeman
 30
 35 MODE 5
 36
 40 REM Initialise and draw maze
 50 GOSUB 1000
 60
 70 REM Make first hole
 80 GOSUB 2000
 90
100 REM Repeat until Quacman through maze
110
120   REM Move Quacman
130   GOSUB 3000
140   IF c<18 THEN GOTO 130
150
155 PRINT TAB(18,r) "Q"
160 PRINT TAB(0,29) "Time taken=";timecount
170
180 END
```

Main program


```
190
200 REM ********* END *********
210
220 REM ***** SUBROUTINES *****
230
990 REM Initialise
1000 LET timecount=0
1010 LET r=0
1020 LET c=0
1030 FOR n=1 TO 20
1040   PRINT " ! ! ! ! ! ! ! ! ! ! "
1050   NEXT n
1055 COLOUR 129
1060 RETURN
1070
1990 REM Make a hole
2000 IF c>16 THEN RETURN
2010 LET h=RND(20)-1
2020 PRINT TAB(c+1,h) " ";
2030 RETURN
2040
2990 REM Move
3000 PRINT TAB(c,r) "Q"
3005 SOUND 1,-10,70,1
3010 FOR t=1 TO 25 : NEXTt
3020 LET timecount=timecount+1
3030 PRINT TAB(c,r) " "
3035 LET key$=INKEY$(5)
3040 IF key$="X" AND r=h THEN LET c=c+2 : GOSUB 2000
3050 IF key$="/" THEN LET r=r+1
3060 IF key$=":" THEN LET r=r-1
3070 IF r<0 THEN LET r=0
3080 IF r>20 THEN LET r=20
3090 RETURN
```

Initialise subroutine

Hole punching subroutine

Movement subroutine

Quacman (Notes)

1–3 These three lines provide a crude timecounter for the program. Each time the program repeats GOSUB 3000, one is added to timecount. The count at the end of the run provides an estimate of your speed, but not a measure of real time. If you would like a real timecounter in the program, you can use TIME:

(a) Change 1000 to 1000 TIME = 0. This sets the computer's timer to zero. (Immediately after TIME = 0 is executed, TIME starts to increase again at 100 units per second.)
(b) Remove line 3020.
(c) Replace line 160 with 160 PRINT TAB (0,29) "Time taken ="; TIME/100"seconds". Note that TIME has to be divided by 100 to give the time in seconds. (TIME is what is called a 'pseudo-variable' – see your User Guide for more details.)

4. We've used a simple maze wall made from the ! character. If you would like a more solid wall, you can create the special character ■ using the method described in 'Monsterzap improved'. To make, say, CHR$224 into ■, add

1025 VDU 23, 224, 255,255,255,255,255,255,255,255

and change line 1040 to

```
1040 PRINT TAB(0,n) " "CHR$224" "CHR$224" "
     CHR$224" "CHR$224" "CHR$224" "CHR$224" "
     CHR$224" "CHR$224" "CHR$224" "
```

5. This repeats GOSUB 3000 (the move routine) until the Quacman has got through the maze. A more elegant method of writing these lines, if you know how to use REPEAT, is

```
REPEAT
GOSUB 3000
UNTIL c>= 18
```

6. Make a beep. If you don't like the sound, experiment a bit until you find one that you like. (The last two numbers in the SOUND statement are the ones to alter.)
7. Another delay loop.
8. This makes the program wait for five hundredths of a second to see whether a key is pressed.
9. Line 3040 moves our 'Q' two columns to the right (i.e. into the next empty column), but only if the 'X' key is being pressed and the 'Q' is opposite the hole.
   We've used key 'X' for 'move right', key '/' for move down or ':' for move up. You may prefer to use others.
10. These two lines make sure that the Quacman doesn't jump out of the top or bottom of the maze. The technique used here is a common trick in programming:
    IF <variable exceeds limit> THEN <variable = limit>
11. Here we have used END to halt the program. STOP and END are almost identical in that they both halt a program. Additionally STOP displays the message 'STOP at line. . .' whereas END does not display a message.

Suggestions for improvement

a. Build in a time limit for getting through.
b. Delete line 3030 and see what happens. How could he leave (webbed) footprints?
c. Make two holes appear in each wall.
d. Then randomly introduce obstacles that delay Quacman's progress.
e. Change the 'Q' to a user defined figure.
f. Give the Quacman an energy quota at the start of the game. Then make the energy run down with passing time. Scatter energy capsules which, if eaten, replace the energy. (If you don't know anything about arrays, you may find it difficult to scatter energy capsules. In that case put them all at a known place e.g. at the tops and bottoms of the columns.)
g. Make something chase the Quacman.
h. Put in a monster or two.

ANAGRAMS 100

```
  10 REM ANAGRAMS 100
  20 REM Copyright Fred Harris
  25 REM Electron version: Richard Freeman
  30
  40 INPUT a$
  50 LET c$=a$
  65
  70 FOR m=1 TO 100
  80   GOSUB 2000 : REM Shuffle
 100   PRINT j$
 110   LET a$=c$
 120   NEXT m
 130
 140 STOP
 150
1900 REM Shuffle
1980 REM ***** SUBROUTINE *****
2000 LET j$=""
2010 FOR k=1 TO LEN c$
2020   LET l=LEN a$
2030   LET n=RND(l)
2040   LET j$=j$+MID$(a$,n,1)
2050   LET a$=LEFT$(a$,n-1)+RIGHT$(a$,l-n)
2060   NEXT k
2070 RETURN
```

Anagrams 100 (Notes)

Anagrams

The single anagram program can be produced from ANAGRAMS 100 by omitting lines 70, 110 and 120.

How the shuffle routine works

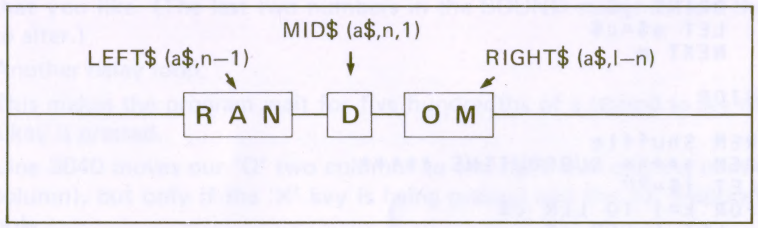
The routine takes letters out of a word and builds a new, shuffled, word out of them. We use

a$	Word to be shuffled
l	Length of a$
j$	New, shuffled word (= "" at start)
n	Position of letter to be picked out of a$.

N.B. Each time we pick a letter out of a$, a$ becomes one letter shorter.

The routine is best understood by an example:

Word to be shuffled	RANDOM
Select a letter at random	e.g. letter 4 (i.e. "D")
Pick the letter out with MID$	MID$ (a$,n,1) picks out "D"
Add the picked letter to j$	j$ = j$ + "D" i.e. j$ = "D"
Take the left part of the old word	LEFT$ (a$,n–1) (i.e. "RAN")
Take the right part of the old word	RIGHT$ (a$,l –n) (i.e. "OM")
Join the left and right parts	a$ = "RAN" + "OM"
Repeat if a$ is not yet empty.	



Suggestions for improvement

a. Try turning this program into a two player version, in which the first player chooses a word and the second has to guess it one letter at a time.

b. When solving crosswords, you usually know where some of the letters are. How could this be incorporated into the program?

c. Usually you are told that your anagram solution will have, say, 3 words and the number of letters in each word. Allow the user to enter both the original anagram and the number of words in the solution and the number of letters in each word. Then adjust the program so that all solutions have the correct format for the solution.
(Hint: the shuffle routine will need to remove all 'spaces' from the shuffled word.)

d. Improve the screen layout to present 10 anagrams at a time neatly placed on the screen with a 'Press SPACE BAR for more' displayed at the bottom.

MATCH

```
 10 REM MATCH
 15 REM Electron version: Richard Freeman
 20 REM Copyright Fred Harris
 30 REM One player version
 40
```



```
 42 MODE 1
 43
 45 REM Initialise
 50 GOSUB 1000
100
105 REM Choose first card
107 REPEAT
110   GOSUB 2000
120   LET firstguess=i : LET n1=n : LET m1=m
123
124  REM Show card
125  GOSUB 3000
130
135  REM Choose second card
137  REPEAT                                    <-④
140    GOSUB 2000
150    UNTIL i<>firstguess
152
154  REM Show card
155  GOSUB 3000
160
170   LET guess=guess+1
180
182  FOR t=1 TO delay : NEXT t
183
185  REM Check for match
190   GOSUB 4000
192                <-⑥
200   IF match=1 THEN GOSUB 5000
202
205   REM If cards do not match
210   IF match=0 THEN GOSUB 6000
220              <-⑦
225   COLOUR 2
230   PRINT TAB(14,19) "TRIES: "; guess
240   PRINT TAB(14,20) "SCORE: "; score
245
250   UNTIL score=10
255
260 FOR z=1 TO 25
262   SOUND 1,-10,7*z,3
264   NEXT z
265
270 END
275 REM ********** END *********
276
278 REM ***** SUBROUTINES *****
290
990 REM Initialise routine
1000 REM
1010 LET guess=0
1015 LET score=0
1020 LET a$="AABBCCDDEEFFGGHHIIJJ"<-②
1030 LET j$=""
1035 LET delay=2500<-①
1040
1050 REM Shuffle
1055 FOR k=1 TO 20 <-⑨
```

```
1060   LET l=LEN a$
1080   LET n=RND(l)
1090   LET j$=j$+MID$(a$,n,1)
1100   LET a$=LEFT$(a$,n-1)+RIGHT$(a$,l-n)
1105   NEXT k
1110
1120 REM Display backs
1130 FOR n=0 TO 4
1140   FOR m=0 TO 3
1150    PRINT TAB(3*n+11,3*m+5); n+5*m+1
1160    NEXT m
1170   NEXT n
1180 RETURN
1185
1990 REM Choose a card
2000 COLOUR 3
2002 REPEAT
2005   REPEAT
2007    PRINT TAB(0,22) "                "
2010    INPUT TAB(0,22) i
2015    i=INT i
2020    UNTIL i>=1 AND i<=20
2022   c$=MID$(j$,i,1) : REM Find chosen card
2023   UNTIL c$<>"_"
2025 PRINT TAB(0,22) "                "
2040 LET m=INT((i-1)/5)
2050 LET n=i-5*m-1
2070 RETURN
2080
2990 REM Showcard
3000 IF c$="A" THEN LET X$="--" : LET y$="--" : COLOUR 1
3010 IF c$="B" THEN LET X$="/\" : LET y$="\/" : COLOUR 2
3020 IF c$="C" THEN LET X$="[]" : LET y$="[]" : COLOUR 3
3030 IF c$="D" THEN LET X$="ıı" : LET y$="ıı" : COLOUR 1
3040 IF c$="E" THEN LET X$="@@" : LET y$="@@" : COLOUR 2
3050 IF c$="F" THEN LET X$="XX" : LET y$="XX" : COLOUR 3
3060 IF c$="G" THEN LET X$="**" : LET y$="**" : COLOUR 1
3070 IF c$="H" THEN LET X$=")(" : LET y$=")(" : COLOUR 2
3080 IF c$="I" THEN LET X$="==" : LET y$="==" : COLOUR 3
3090 IF c$="J" THEN LET X$="00" : LET y$="00" : COLOUR 1
3095 PRINT TAB(3*n+11,3*m+5) x$
3100 PRINT TAB(3*n+11,3*m+6) y$
3110 RETURN
3120
3990 REM Check for match
4000 LET match=0
4010 IF MID$(j$,firstguess,1)=MID$(j$,i,1) THEN LET match =1
4020 RETURN
4030
4990 REM If cards do match
5000 LET j$=LEFT$(j$,firstguess-1)+"_"+RIGHT$(j$,LEN j$-
firstguess)
5010 LET j$=LEFT$(j$,i-1)+"_"+RIGHT$(j$,LEN j$-i)
5020 LET score=score+1
5030 FOR z=53 TO 63
5040   SOUND 1,-10,z*5,1
5050   NEXT z
5060 RETURN
```



```
5070
5990 REM If cards do not match
6000 FOR z=15 TO 1 STEP-1
6010 SOUND 1,-10,75,1
6020   NEXT
6025 COLOUR 2
6040 PRINT TAB(3*n1+11,3*m1+5); firstguess " "
6050 PRINT TAB(3*n1+11,3*m1+6) " "
6060 PRINT TAB(3*n+11,3*m+5); i " "
6070 PRINT TAB(3*n+11,3*m+6) " "
6080 RETURN
```

Match (Notes)

(For a note on the maths of this program, see the notes on the Spectrum version.)

1. 'delay' controls how long the cards are displayed for after an incorrect guess. Increase 'delay' if you want them displayed for a longer period of time.
2. These are the labels for the cards before they are shuffled.
3. The input routine has to be fairly complex because it has to do four things:
   - 3a. Wipe out any previous input display.
   - 3b. Ensure that only whole numbers are entered. There are many ways of doing this. The one that we have used here is

INPUT i	Take in a number
i = INT i	Change it to a whole number

   - 3c. Make sure that the whole number is between 1 and 20.
   - 3d. c$ is the name we give to the letter that stands for the chosen card – that is letter number i in j$. Lines 5000 to 5010 replace each paired letter with "_". This stops you choosing a card that is already matched.
4. Notice also, that the loop at lines 137–150 (4) is also checking the input since we have to check that the second card chosen is not the same as the first card.

5–7 Flags are used for sending information from one part of a program to another. Here the flag 'match' is set to 0 before we check for a match. If a match is found, 'match' is set to 1. 'match' is then used to direct the program to the right choice of subroutine.

8. This is the line where the program checks for a match by comparing the two letters which correspond to the two MID$. Remember that the computer doesn't care about the pictures.
9. Lines 1050–1105 are the shuffle routine from ANAGRAMS 100.
10. The player enters the number of the card that he wants to turn over (1 to 20). Line 2022 finds which letter that card is by selecting it from j$.

Suggestions for improvement

a. Develop user defined characters for the cards.

b. How could this be changed to a two player version, or even to a version for younger children (remember you will have to simplify the INPUT routine).

FIND THE NUMBERS

```
  10 REM FIND THE NUMBERS
  20 REM Copyright Fred Harris
  25 REM Electron version: Richard Freeman
  30
  35 MODE 6
  40 REM Initialise
  50 GOSUB 1000
  60
  70 REM Shuffle number
  80 GOSUB 2000
  90
 100 LET m$=LEFT$(j$,4)
 110
 120 REM Instructions
 130 GOSUB 3000
 140
 150 REPEAT
 160
 165   LET ok=0 <-- (1)
 170   REM Enter guess
 180   GOSUB 4000
 190                  <-- (2)
 200   REM Mark guess
 210   GOSUB 5000
 220
 230   UNTIL ok<>0
 240
 250 REM Result
 260 GOSUB 6000
 270
 280 END
 290
 300 REM ********** END *********
 310
 320 REM ***** SUBROUTINES *****
 330
 990 REM Initialise
1000 LET guess=0
1010 LET a$="1234567890"
1020 LET c$=a$
1030 CLS
1040 RETURN
1050
1990 REM Shuffle
2000 LET j$=""
2010 FOR k=1 TO LEN c$
2020   LET l=LEN a$
2030   LET n=RND(l)
2040   LET j$=j$+MID$(a$,n,1)
2050   LET a$=LEFT$(a$,n-1)+RIGHT$(a$,l-n-1)
2060   NEXT k
2070 RETURN
2080
2990 REM Instructions
3000 PRINT "YOU MUST GUESS THE CODE BY"
3010 PRINT "ENTERING A FOUR DIGIT NUMBER"
```



```
3020 PRINT "(0 TO 9)"
3030 PRINT:PRINT "I WILL MARK AS FOLLOWS:"
3040 PRINT "* MEANS A NUMBER IN WRONG PLACE"
3050 PRINT "+ MEANS A NUMBER IN RIGHT PLACE"
3060 PRINT TAB(0,15) "PRESS A KEY WHEN YOU ARE READY."
3070 d$=INKEY$(1000)
3080 CLS
3090 RETURN
3100
3990 REM Enter guess
4000 REPEAT
4005   INPUT TAB(6,3+guess) g$
4010   IF LEN g$<>4 THEN PRINT TAB(6,3+guess) "A FOUR DIGIT
NUMBER":FOR t=1 TO 1000:NEXT t
4012   PRINT TAB(0,3+guess) "                    "
4015   UNTIL LEN g$=4
4020 LET guess=guess+1
4030 PRINT TAB(6,2+guess) g$
4040 PRINT TAB(15,2+guess);
4050 RETURN
4060                          (3)          (4)
4990 REM Mark
5000 FOR n=1 TO 4
5010   IF MID$(g$,n,1)=MID$(m$,n,1) THEN PRINT "+";
5020   NEXT n
5030 PRINT TAB(19, 2+guess);
5040 FOR n=1 TO 4
5050   FOR m=1 TO 4                    <-- (5)
5060     IF MID$(g$,n,1)=MID$(m$,m,1) AND n<>m THEN PRINT "*";
5070     NEXT m
5080   NEXT n
5090 IF g$=m$ THEN LET ok=1
5100 RETURN
5110
5900 REM Result
6000 FOR n=1 TO 15
6010   SOUND 1,-10,5*n,1
6020   NEXT n
6030 PRINT TAB(0,20) "GOT IT IN ";guess
6040 RETURN
```

Find the Numbers (Notes)

1. Here 'ok' is a flag. As long as ok is 0, the GUESS and MARK GUESS loop (lines 150 to 230) is repeated. But, if the player gets the right answer, the mark routine sets ok to 1 (line 5090). This then allows exit from the repeat loop at line 230 so bringing the result into action (line 260).
2. This repeat loop is designed to ensure that the player enters a four character guess. You can't escape from it until your input has the right length. It is an example of a very common input method of the form:

   REPEAT
   Input
   UNTIL <input satisfies program criteria>

3–5 The marking routine is a bit tricky.

First (3) we have to print a '+' for each correct digit in the correct place in the guess.

Then (4) we have to search for correct digits in incorrect places and print a '*' each time we find one. Notice 'ANDn <>m' (5) which makes sure that we don't print a '*' where a correct digit is in its correct place.

Suggested improvements

a. Make it possible to vary the difficulty of the game by making the number of digits in the number to be guessed a variable.

b. Produce a simple version for children with four coloured objects instead of digits.

c. Add a timer.

d. Improve the screen layout to include instructions at the bottom of the screen, a heading and a more interesting display of the guesses and responses.

Spectrum Programs

MONSTERZAP CORE

```
 10 REM MONSTERZAP
 12 REM CORE LISTING
 20 REM © Fred Harris
 30
 40 REM Initialise                                          ┐
 50 GO SUB 1000                                             │
 60                                                         │
100 REM Draw scene                                          │
110 GO SUB 3000                                             │
120                                                         │
150 REM Main movement loop                                  │
160 FOR c=31 TO -31 STEP -1 <------(2)                      │
170   PRINT AT r,c;"*"                                      │
180   FOR t=0 TO 5: NEXT t <------(1)                      │
190   IF INKEY$="f" OR INKEY$="F" THEN GO SUB 5000          │
200   PRINT AT r,c;" "      <------(3)                      │
210   NEXT c                                               │
220 GO TO 160: REM repeat main loop                         │
230                                                         │
240 STOP                                                    ┘
250
 990 REM Initialise                                         ┐
1000 LET r=0                                                │
1070 LET f=0                                                │
1080 RETURN                                                 ┘
1090
2990 REM Draw scene                                         ┐
3000 PRINT AT 10,0;"HH      HH      HH      HH      HH"     │
3010 PRINT AT 11,0;"HH \   / HH \   / HH \   / HH \   / HH" │
3020 PRINT AT 12,0;"HH 0 ■ 0 HH 0 ■ 0 HH 0 ■ 0 HH 0 ■ 0 HH" │
3030 PRINT AT 13,0;"HH _/\_ HH _/\_ HH _/\_ HH _/\_ HH"     │
3410 PRINT AT 18,0;"0 SHOTS USED"                           │
3420 RETURN                                                 ┘
4990 REM Fire                                               ┐
5000 FOR l=10 TO 13                                         │
5040   PRINT AT l,c;" "                                     │
5050   NEXT l                                              │
5080 LET f=f+1                                              │
5090 PRINT AT 18,0;f                                        │
6000 IF f=40 THEN STOP                                      │
6010 RETURN                                                 ┘
6020
```

Main program

Initialise subroutine

Draw scene subroutine

Zap subroutine

Monsterzap Core (notes)

1. FOR t = 0 TO 5 : NEXT t is the simplest way of inserting a delay into a program. For a longer delay, increase 5; for a shorter delay, decrease it.
2. 'c' stands for column. In this loop, varying c moves the zapper back and forth across the top line of the screen.

3. INKEY$ takes a single keystroke from the keyboard – if there happens to be one. Unlike 'INPUT', INKEY$ does not *wait* for input. If no key is pressed, INKEY$ allows the program to move on to the next line. The total effect of this line is to ensure that the fire routine (line 5000) is only brought into action when 'F' or 'f' is pressed.

Suggestions for extending the program
See 'Monsterzap improved'.

MONSTERZAP IMPROVED

```
  10 REM MONSTERZAP IMPROVED
  12 REM VERSION 2
  20 REM © Fred Harris
  30
  40 REM Initialise
  50 GO SUB 1000
  60
  70 REM Instructions
  80 GO SUB 2000
  90
 100 REM Draw scene
 110 GO SUB 3000
 120
 150 REM Main movement loop
 160 FOR n=31 TO -31 STEP -1
 170   PRINT AT 0,n; "⊠"
 180   FOR t=0 TO 5: NEXT t
 190   IF INKEY$="f" OR INKEY$="F" THEN GO SUB 5000
 200   PRINT AT 0,n;" "
 210   NEXT n
 220 GO TO 160: REM repeat main loop
 230
 240 STOP
 250
 990 REM Initialise
1000 FOR n=0 TO 31
1010   READ g
1020   POKE USR "a"+n,g
1030   NEXT n
1040 INK 0
1050 BORDER 5: PAPER 5
1060
1070 LET f=0
1080 RETURN
1090
1990 REM Instructions
2000 PRINT AT 3,7;"  MONSTERZAP"
2010 PRINT AT 5,7;"PRESS F TO FIRE";AT 6,7;"-ONLY 40 SHOTS!"
2050 PAUSE 100
2060 CLS
2070 RETURN
2080
2990 REM Draw scene
```



```
3000 REM Ground
3010 FOR n=0 TO 14
3020   PRINT PAPER 5;"◄──────③──────►"
3030   NEXT n
3040
3050 REM Sky and stars
3060 FOR n=1 TO 3
3070   PRINT PAPER 4;"◄──────③──────►"
3080   NEXT n
3090
3100 FOR n=1 TO 50
3110     PLOT INK 7;RND*250,RND*70+80 ◄──④
3120   NEXT n
3130
3150
3160 REM Skyscraper
3170 FOR n=0 TO 4
3180   FOR l=10 TO 14
3190       PRINT AT l,7*n+1; INK1;"▀▀"
3200       NEXT l
3310
3320   IF n=4 THEN GO TO 3410
3330
3340   REM Monsters
3350   INK 0: PAPER 8
3360   PRINT AT 12,7*n+3;" \ / "
3370   PRINT AT 13,7*n+3;"*"; INVERSE 1;"0"; INVERSE 0;
" ▀ "; INVERSE 1; "0"; INVERSE 0; "*"
3380   PRINT AT 14,7*n+3;"     "
3390   NEXT n
3400 INK 2
3410 PRINT AT 18,0;"0 SHOTS USED"
3420 RETURN
3990 REM Fire
4990 REM Hit
5000 FOR l=10 TO 13
5010  BEEP .02,2*l
5020  PRINT AT l,n;"*"
5030  PRINT AT l,n;"  "
5040  PRINT AT l,n;" "
5050  NEXT l
5060 PRINT AT 14,n; INK 8;" ▙ "
5070 BEEP .04+.4*(ATTR (l,n)=41),0-20*(ATTR (l,n)=41)
5080 LET f=f+1
5090 PRINT AT 18,0;f
6000 IF f=40 THEN STOP
6010 RETURN
6020
9000 DATA 255,231,231,231,255,231,255,255
9010 DATA 129,219,165,153,153,165,129,36
9020 DATA 36,0,36,255,0,0,0,0
9210 DATA 0,0,0,0,0,133,137,255
```

Monsterzap Improved (Notes)

1&2 These lines are used to set up the special characters used by this program.

These are:

Graphics 'a' as

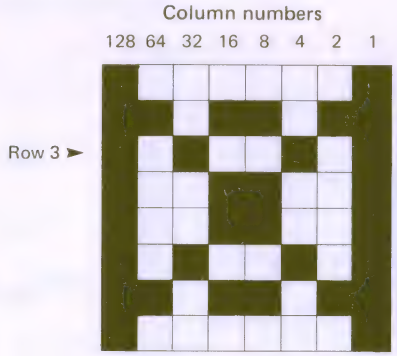
Graphics 'b' as

Graphics 'c' as

Graphics 'd' as

As an example, here is how we created the zapper:

(a) Draw it as blobs on an 8 x 8 grid:



(b) Number the columns, *working right to left,* as 1, 2, 4, 8, 16, 32, 64 and 128 as in the figure above.

(c) For each row, add up the column numbers of the blocks done in columns.

e.g. row 3 128 + 32 + 4 + 1 = 165

(d) Put these row numbers into data statements in your program. (i.e. 8 numbers per special character.)

(e) Then make your program read the characters and poke them into a graphics letter location. 'Poke' is to put a number into a memory location of a computer. In this case we want to place the eight numbers 129, 219, 165, 153, 153, 165, 219 and 129 into the area of memory where the computer stores user defined graphics. We don't need to know where this is, the user statement (followed by the letter we have chosen for our character) automatically uses the right area of memory. Here we make graphics 'b' into the zapper:

e.g.

```
 1 REM How to create a special character
10 FOR i=0 TO 7
20 READ A
30 POKE USR "a"+i,n
```



```
40 NEXT i
45 PRINT "≡"
50 STOP
60 DATA 255,0,255,0,255,0,255,0
```

would set up graphics 'a' as the special character.

≡

To place the character into line 45, type as follows:

45 PRINT" (as usual)

- Hold down SHIFT and press GRAPHICS. You should now get the G cursor.
- Press the letter a.
- Hold down SHIFT and press GRAPHICS. This will cancel graphics.
- Hold down SYMBOL SHIFT and press ".

3. These are strings of 32 spaces. They are used to print a strip right across the screen of the current colour e.g. a strip of sky.
4. This loop plots 50 dots (stars) at random locations.

Suggestions for improvement

a. Arrange for a bomb to fire from the zapper towards the monsters.
b. Build in a time limit.
c. At the end of the game, arrange for the whole cycle to start again at a higher speed.
d. Build in a penalty for hitting the buildings. Make an explosion appear on the screen when a monster is hit. Add a deep beep for hitting a wall.
e. Allow the player to reverse the direction of the zapper. (e.g. Press ENTER to reverse the direction of movement.) Then make the monsters fire back!

QUACMAN IMPROVED

(This program is an extension of the Quacman program shown in the television series. It is basically the same program but with sound and colour added.)

```
10 QUACMAN IMPROVED
20 REM © Fred Harris
30
40 REM Initialise
50 GOSUB 1000
60
70 REM Make first hole
80 GO SUB 2000
90
```

```
100 REM Repeat until done
110
120  REM Move racer
130  GO SUB 3000
140  IF c<31 THEN GOTO 130
150
155 PRINT AT h,31;"Q"
160 PRINT AT 21,0; FLASH 1;"TIME TAKEN=";timecount ---③
170
180 STOP
190
200 REM ********* END *********
210
220 REM ***** SUBROUTINES *****
990 REM Initialise
1000 LET timecount=0 ---①
1010 LET r=0
1020 LET c=0
1030 FOR n=1 TO 20
1040  PRINT " ■■■■■■■■■■■■■■■■"
1050  NEXT n
1060 RETURN
1070
1990 REM Make a hole
2000 IF c>31 THEN RETURN
2010 LET h=INT (RND*20)
2020 PRINT AT h,c+1;" "
2030 RETURN
2040
2990 REM Move
3000 PRINT AT r,c;"Q"
3010 BEEP .02,20-ABS (r-h)
3020 LET timecount=timecount+1 ---②
3030 PRINT AT r,c; PAPER 6;" "
3040 IF INKEY$="8" AND r=h THEN LET c=c+2  : GO SUB 2000
3050 IF INKEY$="7" THEN LET r=r-1
3060 IF INKEY$="6" THEN LET r=r+1
3070 IF r<0 THEN LET r=0  ]---④
3080 IF r>20 THEN LET r=20]
3090 RETURN
```

Quacman Improved (Notes)

1–3 These three lines provide a crude timecounter. Each time that the 'move racer' subroutine is executed, 'timecount' is increased by 1. The final value of 'timecount' is displayed when you get through the maze.

(Providing an accurate clock on the ZX Spectrum is very difficult and uses advanced programming techniques.)

4. These two lines make sure that the Quacman doesn't jump out of the top or bottom of the maze. The technique used here is a common trick in programming:

IF <variable exceeds limit> THEN <variable = limit>

Suggestions for improvement

See Electron suggestions.

ANAGRAMS CORE

```
 10 REM ANAGRAMS
 20 REM © Fred Harris
 30
 40 INPUT a$
 65
 80   GO SUB 2000 : REM Shuffle word
100   PRINT j$
130
190 STOP
200
1990 REM Shuffle word
2000 LET j$=""
2010 FOR k=1 TO LEN a$
2020  LET l=LEN a$
2030  LET n=INT (RND*l)+1
2040  LET j$=j$+a$(n)
2050  LET a$=a$( TO n-1)+a$(n+1 TO )
2060  NEXT k
2070 RETURN
```

ANAGRAMS 100

```
 10 REM ANAGRAMS 100
 20 REM © Fred Harris
 30
 40 INPUT a$
 50 LET c$=a$
 65
 70 FOR m=1 TO 100
 80   GO SUB 2000 : REM Shuffle word
100   PRINT j$
110   LET a$=c$
120   NEXT m
130
190 STOP
200
1990 REM Shuffle word
2000 LET j$=""
2010 FOR k=1 to LEN c$
2020  LET l=LEN a$
2030  LET n=INT (RND*l)+1
2040  LET j$=j$+a$(n)
2050  LET a$=a$( TO n-1)+a$(n+1 TO )
2060  NEXT k
2070 RETURN
```

MATCH (ONE PLAYER)

```
 10 REM MATCH
 20 REM © Fred Harris
```

```
 30 REM ONE PLAYER VERSION
 40
 45 REM Initialise
 50 GO SUB 1000
100
105 REM Choose first card
110 GO SUB 2000
120 LET firstguess=i : LET n1=n : LET m1=m
124 REM Show card
125 GO SUB 3000
130
135 REM Choose second card
140 GO SUB 2000
150 IF i=firstguess THEN GO TO 140
154 REM Show card
155 GO SUB 3000
160
170 LET guess=guess+1
180
182 PAUSE 50
185 REM Check for match
190 GO SUB 4000
195 REM Match action
200 IF match=1 THEN GO SUB 5000
205 REM No match action
210 IF match=0 THEN GO SUB 6000
220
230 PRINT AT 19,8;"TRIES:";guess
240 PRINT AT 20,8;"SCORE:";score
250 IF score<10 THEN GO TO 110
260 BEEP .5,0: BEEP .5,4: BEEP .5,7: BEEP 1,12
265
270 STOP
275 REM ******** END *********
276 REM **********************
277
278 REM ****** SUBROUTINES ****
290
990 REM Initialise routine
1000 BORDER 5
1010 LET guess=0
1015 LET score=0
1020 LET a$="AABBCCDDEEFFGGHHIIJJ"
1030 LET J$=""
1040
1050 REM Shuffle
1055 FOR k=1 TO 20
1060   LET l=LEN a$
1080   LET n=INT (RND*l)+1
1090   LET j$=j$+a$(n)
1100   LET a$=a$( TO n-1)+a$(n+1 TO )
1105   NEXT k
1110
1120 REM Display backs
1130 FOR n=0 TO 4
1140   FOR m=0 TO 3
1150     PRINT AT 3*m+5,3*n+8;n+5*m+1
1160     NEXT m
```



```
1170   NEXT n
1180 RETURN
1185
1190 REM Choose a card
2000 INPUT i
2010 LET i=INT i
2020 IF i <1 OR i>20 THEN BEEP 1,-10: GO TO 2000
2030 IF j$(i)="-" THEN BEEP .1,-20: GO TO 2000
2040 LET m=INT ((i-1)/5)
2050 LET n=i-5*m-1
2070 RETURN
2080
2990 REM Showcard
3000 IF j$(i)="A" THEN LET x$="▄▄": LET y$="oo": LET colour=1
3010 IF j$(i)="B" THEN LET x$="▙▟": LET y$="▜▛": LET colour=2
3020 IF j$(i)="C" THEN LET x$="▙ ": LET y$="▜ ": LET colour=4
3030 IF j$(i)="D" THEN LET x$="/\": LET y$="\/": LET colour=0
3040 IF j$(i)="E" THEN LET x$="▚▞": LET y$="▚▞": LET colour=3
3050 IF j$(i)="F" THEN LET x$="▆▆": LET y$="▆▆": LET colour=4
3060 IF j$(i)="G" THEN LET x$="**": LET y$="**": LET colour=1
3070 IF j$(i)="H" THEN LET x$="++": LET y$="++": LET colour=2
3080 IF j$(i)="I" THEN LET x$="▛▜": LET y$="▟▙": LET colour=0
3090 IF j$(i)="J" THEN LET x$="▜/": LET y$="/▙": LET colour=4
3100 PRINT INK colour;AT 3*m+5,3*n+8;x$
3110 PRINT INK colour;AT 3*m+6,3*n+8;y$
3120 RETURN
3130
3990 REM Check for match
4000 LET match=0
4010 IF j$(firstguess)=j$(i) THEN LET match=1
4020 RETURN
4030
4990 REM Match action
5000 LET j$(firstguess)="-"
5010 LET j$(i)="-"
5020 LET score=score+1
5030 FOR z=12 TO 24
5040   BEEP .03,z
5050   NEXT z
5060 RETURN
5070
5990 REM No match action
6000 FOR z=12 TO 0 STEP -1
6010   BEEP .03,z
6020   NEXT z
6030 BEEP 1,-20
6040 PRINT AT 3*m1+5,3*n1+8;firstguess;" "
6050 PRINT AT 3*m1+6,3*n1+8;"  "
6060 PRINT AT 3*m+5,3*n+8;i;" "
6070 PRINT AT 3*m+6,3*n+8;"  "
6080 RETURN
```

Match (One Player) — Notes

1. These are the labels for the cards before they are shuffled.
2. The input routine has to be fairly complex because it has to do three things:

2a. Ensure that the entered number is a whole number. INT cuts any decimal number down to the whole number below it. e.g. INT 2.3 is 2.

2b. Make sure that the whole number is between 1 and 20.

2c. Make sure that the card chosen has not already been paired-up.

3. Notice that lines 140 and 150 are also checking the input since we have to ensure that the second card choice is not the same as the first.

4–7 Flags are used for sending information from one part of a program to another. Here the flag 'match' is set to 0 before we check for a match. If a match is found, 'match' is set to 1. 'match' is then used to direct the program to the right choice of subroutine.

8. This is the line where the program checks for a match. Remember that the computer doesn't care about the pictures.

The maths of this program may look rather complex but it's all designed to keep the programming simple. The cards are in five rows and four columns:

Row \ COLUMN	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	1	11	12
3	13	14	15	16
4	17	18	19	20

The rows are numbered 0 to 4 and use the variable 'm' in the program. The columns are numbered 0 to 3 and use the variable 'n' in the program.

When the player picks a card (line 2000), lines 2040 and 2050 work out the values of m and n.

Later, lines 3100 and 3110 work out where to print the card on the screen. (At (3*m+5, 3*n+8) for the top half and at (3*m+6, 3*n+8) for the bottom half.)

And finally, if a pair of cards have to be wiped out and replaced with their numbers, then this is done by lines 6040 and 6060 (replace the numbers) and by lines 6050 and 6070 (wipe out the lower parts of the cards).

Suggestions for improvement

See Electron list.

FIND THE NUMBERS

```
10 REM FIND THE NUMBERS
20 REM © Fred Harris
30
40 REM Initialise
50 GO SUB 1000
60
70 REM Shuffle number
80 GO SUB 2000
90
100 LET m$=j$( TO 4)
110
120 REM Instructions
130 GO SUB 3000
140
150 REM Repeat until correct
160
165   LET ok=0 <----(1)
170   REM Enter guess
180   GO SUB 4000
190
200   REM Mark guess
210   GO SUB 5000
220
230   If ok=0 THEN GO TO 180
240
250 REM Result
260 GO SUB 6000
270
280 STOP
290
300 REM ******** END *********
310
320 REM ***** SUBROUTINES ***
330
990 REM Initialise
1000 LET guess=0
1010 LET a$="1234567890"
1020 LET c$=a$
1030 CLS
1040 RETURN
1050
1990 REM Shuffle
2000 LET j$=""
2010 FOR k=1 TO LEN c$
2020   LET l=LEN a$
2030   LET n=INT (RND*l)+1
2040   LET j$=j$+a$(n)
2050   LET a$=a$ (TO n-1)+a$(n+1 TO )
2060   NEXT k
2070 RETURN
2080
2990 REM Instructions
3000 PRINT "YOU MUST GUESS THE CODE BY"
3010 PRINT "ENTERING A FOUR DIGIT NUMBER"
3020 PRINT "(0 TO 9)"
3030 PRINT : PRINT "I WILL MARK AS FOLLOWS:"
3040 PRINT "* MEANS A NUMBER IN WRONG PLACE"
3050 PRINT "+ MEANS A NUMBER IN RIGHT PLACE"
3060 PRINT AT 15,0;"PRESS A KEY WHEN YOU ARE READY."
3070 PAUSE 500
3080 CLS
3090 RETURN
3100
```

```
3990 REM Enter guess
4000 INPUT g$
4010   IF LEN g$<>4 THEN PRINT AT 0,0;"A FOUR DIGIT NUMBER"
: PAUSE 40: PRINT AT 0,0;"                    ": GO TO 4000
4020 LET guess=guess+1
4030 PRINT AT 2+guess,6;g$
4040 PRINT AT 2+guess,15;
4050 RETURN
4060
4990 REM Mark
5000 FOR n=1 TO 4
5010   IF g$(n)=m$(n) THEN PRINT "+";          (2)
5020   NEXT n
5030 PRINT AT 2+guess,19;
5040 FOR n=1 TO 4                                  (3)
5050   FOR m=1 TO 4
5060     IF g$(n)=m$(m) AND n<>m THEN PRINT "*";   (4)
5070     NEXT m
5080   NEXT n
5090 IF g$=m$ THEN LET ok=1
5100 RETURN
5110
5900 REM Result
6000 FOR n=1 TO 15
6010   BEEP .03,n
6020   NEXT n
6030 PRINT AT 20,0;"GOT IT IN "; guess
6040 RETURN
```

Find the Numbers (Notes)

1. 'ok' is a flag. As long as ok is 0, the enter guess/mark guess loop is repeated. But if the player gets the right answer, the mark routine sets ok to 1 (line 5090). This then allows exit from the loop at line 230.

2–4 The marking routine is a bit tricky.

First (2) we have to print a '+' for each correct digit in the correct place in the guess.

Then (3) we have to search for correct digits in incorrect places and print a '*' each time we find one. Notice 'ANDn<>m' (4) which makes sure that we don't print a '*' where a correct digit is in its correct place.

SPECTRUM STRINGS

An odd feature of the Spectrum computer is that it does not distinguish between string variables in upper case and lower case. So, to the Spectrum, B$ is the same variable as b$. As a result, the Spectrum only has 26 string variables, A$, B$, C$, ... Z$ (or, if you like, a$, b$, c$, ... z$).This restriction prevents you from using meaningful string names (e.g. name$) as you can on the Electron. (There is no apparent reason for this, except that the Spectrum developed out of the ZX81 computer which also had very limited string store facilities.)

REPEAT

BBC BASIC, along with other advanced programming languages provides a REPEAT. . . UNTIL facility. Spectrum BASIC does not provide REPEAT, but it can be simulated.

First, look at how REPEAT. . . UNTIL works. It is used to make a program repeat a section of code until an exit condition is met. A common application is to ensure that only valid information is entered at the keyboard.

```
REPEAT
INPUT "Enter a number from 1 to 3" num
UNTIL num>=1 AND num<=3
```

This will ensure that the program will not exit from the loop until you enter an appropriate number.

To simulate this in Spectrum BASIC, you can use a FOR. . . NEXT. . . loop and then interfere with the loop counter. The following loop

```
FOR i = 0 TO 1
    LET i = 0                                  reset the loop counter to
                                               force the loop to be repeated.
    INPUT "Enter a number from 1 to 3" num
    If num > =1 AND num < =3 THEN LET i=1      If the exit condition is met,
                                               set the loop counter to its
    NEXT i                                     exit value.
```

behaves in exactly the same way as the genuine REPEAT loop above.

ADDRESSES

Commodore Business Machines UK Ltd
(Commodore Information Centre)
675 Ajax Avenue
Slough
Berks
Tel: (0753) 79292

Apple Computers UK Ltd
Eastman
Hemel Hempstead
Herts
Tel: (0442) 60244

Oric Products International Ltd
Coworth Park Mansion
Coworth Park
London Road
Sunninghill
Ascot
Berks
Tel: (0990) 27686

Oric Assembly
Unit 11
Hampton Farm Industrial Estate
Hampton Road West
Hanworth
Middx.
Tel: (01) 755 1133

Dragon Data Ltd
The Kenfig Industrial Estate
Margam
Port Talbot
West Glam.
Tel: (0656) 744700

Acorn Computers Ltd
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN
Tel: (0223) 245200

Sinclair Research Ltd
28 Stanhope Road
Camberley
Surrey
Tel: (0276) 686161

BOOKS

The programs in this leaflet are developed in more detail in:
Paul Shreeve, *Me & My Micro* (National Extension College).

A simple approach to structured programming (the methods used here and in the TV series) can be found in:
Richard Freeman, *Step by Step BASIC* (BBC/Electron edition) (Lifelong Learning Ltd)
Richard Freeman, *Step by Step BASIC* (ZX Spectrum edition) (Lifelong Learning Ltd)

A more advanced course on structured programming can be found in:
Richard Freeman, *Structured BASIC* (BBC/NEC)